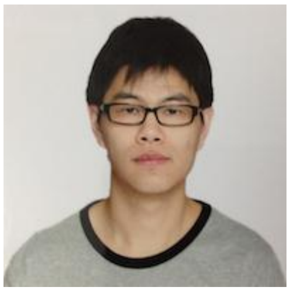
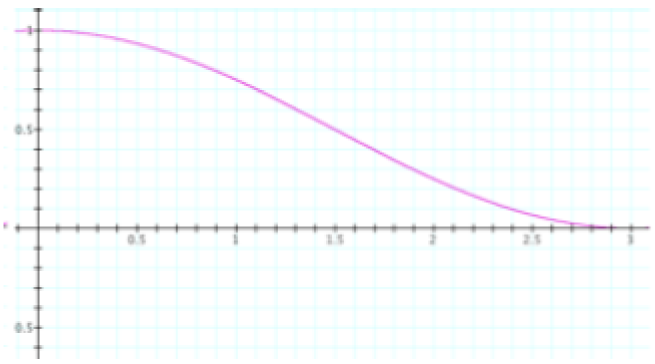
Team Member 1: Yik Wai Ng GTID: 902954691
Team Member 2: Ho Pan Chan GTID: 902956511



Velocity Field Curve:

Particle behavior: we would like the points further away may be less influenced by the velocity field. So, we use the cosine square decay function to determine the influence from the velocity field according to the distance between the particles and the field curve.

v_decay_ratio = cos^2(distance π / longest influencing distance)



```
float decayRatio=0;
if(distanceParticileClosestC<310)
   //cosine square decay function,  decay according to distance from 0 to 300
   decayRatio=sq(cos(distanceParticileClosestC*PI/300));
newVel=V(decayRatio,newVel);
```

Generator:

Since the sphere (generator)'s radius is known, we can use the following transformation to get the random sampling of the surface.

$$r^2 = x^2 + y^2 + z^2; x = r\cos\alpha\cos\beta; y = r\sin\alpha; z = r\cos\alpha\sin\beta;$$

By getting random number for α and β (from 0 to 2π), we can convert it and produce a random sampling of the surface of the sphere.

Gravity:

Discuss how to implement proper gravitational forces

Learnt from high school physics, the gravitational forces between two massed objects are

$F = G\frac{m_1 m_2}{r^2}$, where G is the gravitational constant, m_1 is the first mass, m_2 is the second mass, r is the distance between the centers of the masses. Assume first object is the particle, second object is the obstacle and obstacle does not move at all, then the force acting on the particle:

$$\overrightarrow{F} = m_1 \overrightarrow{a} = G\frac{m_1 m_2}{\|\overrightarrow{r}\|^2}\widehat{r}$$

$$\overrightarrow{a} = \frac{Gm_2}{\|\overrightarrow{r}\|^2}\widehat{r} = \frac{k}{\|\overrightarrow{r}\|^2}\widehat{r}$$, where k is a constant.

By altering the k, one can simulate the strength of force acting on each particle. In each frame rate interval T, the velocity of the particle is in addition affected by the gravitational force, which will pull it towards the obstacle. There is no perfect number that makes the best simulation of the gravity field, but by giving the degree of freedom, users can change it to suit their needs.

Collision detection

We first update all particles' velocities, and then we split into two computations. First one is to check the collision between particle pairs. For each pairs of particles, we assume they will collide after time s. Then we solve the equation d(A+sU,B+sV)=a+b, where A, B are the centers of the particles, U, V are their corresponding velocities and a, b are the radius of them. For all of the particles pairs, we find the pair with the smallest positive collision time ct_1 and store it.

Second one is to check the collision between particles and obstacle. Like checking particles collision, we solve the same equation with B is the obstacle. And then we find the smallest positive collision time ct_2 and store it. Then we compare ct_1 and ct_2 to get the smaller one and use it to do the collision action.

Collision reaction:

A) Collide with massive obstacle

Since we assume the obstacle is of infinite mass so simply reverse the normal component of the velocity of the particle

$N = C_1C_2$.unit (normal direction to both at contact point)

$U = (V \bullet N)\, N$ (normal components of V)

$V' = V_1 - 2U$ (swap direction of U)

B) Collide with other particles

Since we assume all the particles are of the same mass, so the change of velocity is simply the exchange of their normal velocity

$N = U(C_1C_2)$ (normal direction to both at contact point)

$U_1 = (V_1 \bullet N)\ N$ (normal components of velocities)

$U_2 = (V_2 \bullet N)\ N$

$V_1' = V_1 - U_1 + U_2$ (cancel U_1 and add U_2)

$V_2' = V_2 - U_2 + U_1$ (to exchange their normal velocities)

Problems regarding collision detection and reaction:

1) **The local velocity field is too strong,** causing some abnormal collision. For example, there may be a lot of subtle collisions between two particles (they are so near and move along the velocity field together)/ between particle and obstacle, slow down the program. Whenever there is an update on the particles' velocity according to the field velocity (after they advance to next points), the change in velocity caused by collision seem to be "erased" (not significant to observe)
   - To make the effect of collision more obvious, we can change the blend parameter to smaller value (e.g. 0.1) so that the influence of the field velocity is less significant, and the collision behavior is comparably more obvious.
2) Since we assume at most one collision happens at any given time t, we **may miss some collisions** (though the probability is low). For example, two pairs (A B and C D) of particles collide at the same, then we advance by t1 (first collision t), then collision of other pair will be missed.
   - Cannot be solved && is assumption
3) **Collision between more than 2 particles is not handled**
   - Too complicated and cannot be solved
4) **Floating point rounding errors**: when the collision time or other things calculated is very small, it is more likely to subject to floating point rounding errors.
   - Cannot be solved

** The number of particles that the system is no longer capable of providing 30 fps: ~200 particles

This number is regarding to the power and configuration of the computer system. Also, the 3D graphics details requirement such as sphere details has significant impact on the fps.